Unix and Knoppix

After its birth and up until AT& T's Version 6 Unix had been very useful in Computer Science courses on Operating Systems since the source code could be freely distributed, discussed and taught. With the arrival of Version 7, arguably the best one ever, the availability of the code was discontinued. This led to Andrew Tanenbaum who was a professor teaching Computer Science in the Netherlands, writing from scratch in 1987 his own code in C for Minix (Mini Unix). Having written a book in which he gave all the code for the new operating system he used it for his own teaching and research and, together with his research students produced Minix 2 in 1997 and Minix 3 in 2004. (Tanenbaum and Woodhull, 2006)

The purpose of Minix was to demonstrate how an operating system work. A computer science student whose name was Linus Torvalds began first by using Minix and writing his own drivers and so on, and then decided to write his own code for an operating system to do what Unix does and to be more efficient than Minix. His aim was not for educational purpose but for real use. Thus came Linux 0.01 in August 1991 and Linux 1.0 on 13 March 1994.

Linux work force has formed into various groups, to mention but two Debian and Red Hat. All in Unix family, including Minix and Linux, follow the standard set up by IEEE called POSIX (Portable Operating System Interface).

Knoppix is a live-CD system based on Debian. Here we chose Knoppix 3.9 because it is efficient and has both `emacs` and `gcc` needed for the course. The Knoppix CD can also be installed on to a computer. You would only do this if you want to replace all the existing systems on your computer with Knoppix's Debian.

Emacs

The programme Editor Macros (Emacs) was written by Richard Stallman. It was him who founded both the Free Software Foundation and the GNU project. Emacs commands in normal use are in the form of key sequences. All of these commands are actually short cuts of some full-text commands, which are seldom used except in cases where no equivalent key sequence exists. One also finds combinations of key sequences and full-text commands, for example `M-x replace-string`.

Tabble 1 gives some of the key sequences in common use. Here the prefixes `C` and `M` means respectively a control key and a meta (escape) key. With the key sequences like `C-x C-f` for openning a file the control key can be pressed and kept down while the other two keys are pressed one after the other. While the control key is kept down, however, the meta key is released before its partner key is pressed, for example `M-v` for scrolling up is `M` first and then `v`.

Function	*Key sequence*	*Description*
Cursor	`C-a`	cursor to beginning of the line
	`C-b`	move cursor left
	`C-e`	cursor to end of the line
	`C-f`	move cursor right
	`C-n`	move cursor down
	`C-p`	move cursor up
	`M-b`	move backwards one word
	`M-f`	move forwards one word

Table 1 *Emacs key sequences*

Function	*Key sequence*	*Description*
Edit	C-d	delete a character
	C-k	erase to end of the line
	C-t	swap two adjacent letters
	C-x C-i	read in the contents of a file
	C-x C-r	reread a file
	C-x C-t	swap two lines
	C-o	insert a line at cursor position
	C-w	delete to mark
	C-y	yank
	C-_	undo the previous change
	M-c	capitalise initial letter of the word
	M-d	delete a word
	M-l	lowercase to end of word
	M-t	swap two words
	M-u	uppercase to end of word
	M-%	query replace
Marking	C-Spc	set mark
Programme	C-g	quitting the current action in preparation
	C-x C-c	exit from the programme
	C-x C-f	open a file
	C-u M-!	enter shell commands, output into this buffer
	M-!	enter shell commands, output to another buffer
Saving	C-x C-s	save the current buffer
	C-x C-w	save current buffer to a specified file
Scrolling	C-l	scroll to put current cursor position in middle
	C-x [	scroll backwards one page
	C-x]	scroll forwards one page
	C-v	scroll down one screen
	M-v	scroll up one screen
	M-<	goto beginning of the file
	M->	goto end of the file
Search	C-r	search backwards for a specified string
	C-s	search forwards for a specified string

Table 1(continued) *Emacs key sequences*

The sequence `M-\%` queries and replaces, while `M-x replace-string` replaces all instances met without asking for confirmation. While replacing thus `y` does the replacing and continues to the next occurrence while `n` does no replacing and continue. Also, `!` can replace the rest without asking, `?` gives a list of options, `.` replaces the current instance and quits, `,` replaces the current instance but does not move on to the next one.

Knoppix

The philosophy of Knoppix is to allow as little write access as possible. But one can issue the command `su` and then mount a floppy drive using the following command (assuming the drive is logically on /dev/fd0).

```
mount -t msdos -w /dev/fd0 /floppy
```

Similarly one may also mount a hard disk or a partition using, for example,

```
mount -w /mnt/hda5
```

Here we have prior to this checked (by examining the device icon on desktop) that the partition we want to work on is `/mnt/hda5`.

Unix

Table 2 lists some of the Unix commands that are frequently used together with their description. To read the manual page for a command, type `man` followed by the command's name. Typine a command's name followed by `--help` gives a brief description of the options available. To see what `man` does, say, type `man man`. Most relevant to this course, study `man emacs` and `man gcc` to learn more about both `emacs` and `gcc` we use.

With Unix everything is represented as a file. This includes things like directories and devices.

Command	*Description*
`awk`	pattern scanning and processing language
`bc`	a calculator language
`cal`	a calendar
`calendar`	reminder service
`cat`	concatenate files and print on standard output
`cp`	copy files
`cut`	removes sections from each line of a file
`dc`	a calculator in Reverse Polish Notation
`df`	reports file system disk space usage
`dmesg`	print or control the kernel ring buffer
`du`	gives memory usage of each file
`emacs`	an extensible editor
`find`	search for files
`grep`	prints lines matching a pattern
`less`	file perusal filter for viewing on CRT
`ls`	list directory contents
`man`	on-line reference manual
`mkdir`	make a directories
`more`	file perusal filter for viewing on CRT
`mount`	mount a file system
`mv`	move files
`nl`	number lines in files
`rmdir`	remove empty directories
`sed`	a stream editor
`sort`	sort lines of text files
`su`	change UID to that of the superuser (root) or another user
`tar`	archives files
`tee`	reads from standard input, write to standard output and files
`touch`	create a blank file
`umount`	unmount a file system
`uniq`	removes duplicate lines from a sorted file
`vi`	basic text editor
`wc`	gives the number of newlines, words and bytes in files

Table 2 *Unix commands*

A *regular expression* is an expression that uses special characters as masks for file names. Thus the result is a set of a number of files which meet the masking. For example,

```
ls lond06a0{0[1-9],[1-6][0-9]}.jpg
```

will list all the files from `lond06a001.jpg` to `lond06a069.jpg` whereas,

```
ls lond*.jpg
```

will list all files beginning with `lond` and ending with `.jpg`.

The text editor `vi` is basic, which means that it is very important to learn how to use. Likelier than not, in dire situations when everything else fails it would be the only thing to work. So it is essential for a programmer to know.

As an example of `awk` try the following.

```
awk -F: '{print $5 $3}' /etc/passwd
```

`dmesg` is used to list the boot messages. We can redirect the output of this command into a file by `dmesg >` *file*.

Piping allows one process to transfer its output to another. The symbol for a pipe is simply a vertical bar. The following finds all `.txt` files whose name contains the string "cpg".

```
find ./ *.txt  grep cpg
```

Some command line tips, `C-p` brings up the previous command entered, `C-n` brings up the following command, and `C-u` clear to beginning of the line.

Bibliography

Jack Tackett, Jr and Steven Burnett. *Special Edition Using Linux.* 4th ed., Que, 1999

Andrew S Tanenbaum and Albert S Woodhull. *Operating Systems Design and Implementation.* 3rd ed., Prentice-Hall, 2006